THIRD ORDER NEWTON'S METHOD FOR ZERNIKE POLYNOMIAL ZEROS

RICHARD J. MATHAR

Abstract. The Zernike radial polynomials are a system of orthogonal polynomials over the unit interval with weight x. They are used as basis functions in optics to expand fields over the cross section of circular pupils. To calculate the roots of Zernike polynomials, we optimize the generic iterative numerical Newton's Method that iterates on zeros of functions with third order convergence. The technique is based on rewriting the polynomials as Gauss hypergeometric functions, reduction of second order derivatives to first order derivatives, and evaluation of some ratios of derivatives by terminating continued fractions.

A PARI program and a short table of zeros complete up to polynomials of 20th order are included.

1. Classical Orthogonal Polynomials: Hofsommer's Newton Method

The generic third order Newton's Method to compute roots $f(x) = 0$ numerically improves solutions $x_i \to x_{i+1} = x_i + \Delta x$ iteratively, starting from initial guesses, via computation of corrections

$$\Delta x = -\frac{f(x)}{f'(x)} \Big/ \left(1 - \frac{f(x)}{2f'(x)}\,\frac{f''(x)}{f'(x)}\right) \tag{1}$$

where $f(x)$, $f'(x)$ and $f''(x)$ are the function and its first and second derivatives at the current best approximation x_i [8, 11, 14]. For some classes of orthogonal polynomials, f''/f' can be derived from f/f' [12, 24], which means the update can be done to third order at essentially no additional numerical expense. If we divide the differential equation of the classical orthogonal polynomials, for example as tabulated in [1, 22.6][17],

$$h_2(x)f'' + h_1(x)f' + h_0(x)f = 0, \tag{2}$$

through f', (1) turns into

$$\Delta x = -\frac{f(x)}{f'(x)} \Big/ \left[1 + \frac{1}{2h_2(x)}\frac{f(x)}{f'(x)}\left(h_0(x)\frac{f(x)}{f'(x)} + h_1(x)\right)\right]. \tag{3}$$

Date: February 1, 2008.

2000 *Mathematics Subject Classification.* Primary 26C10, 33C45; Secondary 78M25.

Key words and phrases. Zernike Polynomial, Jacobi Polynomial, circular pupil, root finding, Newton Method.

Supported by the NWO VICI grant 639.043.201 to A. Quirrenbach, "Optical Interferometry: A new Method for Studies of Extrasolar Planets."

Structure relations [18] relate the ratio f/f' to ratios at shifted indices n as tabulated for example in [1, 22.8],

$$g_2(x)f'_n(x) = g_1(x)f_n(x) + g_0(x)f_{n-1}(x); \tag{4}$$

$$\Rightarrow \quad \frac{f_n(x)}{f'_n(x)} = \frac{g_2(x)}{g_1(x) + g_0(x)\frac{f_{n-1}(x)}{f_n(x)}}. \tag{5}$$

The benefit is that the three-term recurrence equations, in the notation of [1, 22.7]

$$a_{1,n-1}f_n(x) = (a_{2,n-1} + a_{3,n-1}x)f_{n-1}(x) - a_{4,n-1}f_{n-2}(x), \tag{6}$$

lead to terminating continued fraction representations for f/f'

$$\frac{f_{n-1}(x)}{f_n(x)} = \frac{a_{1,n-1}}{a_{2,n-1} + a_{3,n-1}x - a_{4,n-1}\frac{f_{n-2}(x)}{f_{n-1}(x)}}. \tag{7}$$

This is recursively inserted into the denominator of (5) to lower the index n until f_0/f_1 is reached, which avoids problems with cancellation of digits.

This work here implements this strategy for the family of Zernike polynomials, $f = R_n^m$, namely (i) fast calculation of f''/f' from f/f', (ii) calculation of f/f' from terminating continued fractions, both without evaluation of f or its derivatives via direct methods like Horner schemes.

2. Zernike Polynomials: Derivatives and Roots

2.1. **Definition.**

We define Zernike radial polynomials in Noll's nomenclature [21, 23, 15, 26, 2, 25] for $n \ge 0$, $n - m = 0 \mod 2$, $m \le n$ as

$$R_n^m(x) = \sum_{s=0}^{(n-m)/2} \frac{(-1)^s (n-s)!}{s![(n+m)/2 - s]![(n-m)/2 - s]!} x^{n-2s} \tag{8}$$

$$= \sum_{s=0}^{(n-m)/2} (-1)^s \binom{n-s}{s} \binom{n-2s}{(n-m)/2 - s} x^{n-2s}. \tag{9}$$

Following the original notation, we will not put the upper index m in R_n^m—which is not a power—into parentheses. The normalization integral is

$$\int_0^1 x\, R_n^m(x) R_{n'}^m(x) dx = \frac{1}{2(n+1)} \delta_{n,n'}. \tag{10}$$

The complete set of examples for $n \le 4$ is

$$R_0^0(x) = 1; \qquad R_1^1(x) = x; \tag{11}$$

$$R_2^0(x) = 2x^2 - 1; \qquad R_2^2(x) = x^2; \tag{12}$$

$$R_3^1(x) = 3x^3 - 2x; \qquad R_3^3(x) = x^3; \tag{13}$$

$$R_4^0(x) = 6x^4 - 6x^2 + 1; \qquad R_4^2(x) = 4x^4 - 3x^2; \qquad R_4^4(x) = x^4. \tag{14}$$

The inversion of (9) decomposes powers x^i into sums of $R_n^m(x)$, ($i \ge m$, $i - m$ even),

$$x^i \equiv \sum_{n=m \mod 2}^{i} h_{i,n,m} R_n^m(x); \quad i - m = 0, 2, 4, 6, \ldots; \tag{15}$$

$$h_{i,n,m} = 2(n+1) \sum_{s=0}^{(n-m)/2} \frac{(-1)^s}{n - 2s + i + 2} \binom{n-s}{s} \binom{n-2s}{(n-m)/2 - s}. \tag{16}$$

The basic examples are

$m = 0$:

$$x^0 = R_0^0(x); \tag{17}$$

$$x^2 = \frac{1}{2}R_0^0(x) + \frac{1}{2}R_2^0(x); \tag{18}$$

$$x^4 = \frac{1}{3}R_0^0(x) + \frac{1}{2}R_2^0(x) + \frac{1}{6}R_4^0(x); \tag{19}$$

$$x^6 = \frac{1}{4}R_0^0(x) + \frac{9}{20}R_2^0(x) + \frac{1}{4}R_4^0(x) + \frac{1}{20}R_6^0(x); \tag{20}$$

$m = 1$:

$$x = R_1^1(x); \tag{21}$$

$$x^3 = \frac{2}{3}R_1^1(x) + \frac{1}{3}R_3^1(x); \tag{22}$$

$$x^5 = \frac{1}{2}R_1^1(x) + \frac{2}{5}R_3^1(x) + \frac{1}{10}R_5^1(x); \tag{23}$$

$$x^7 = \frac{2}{5}R_1^1(x) + \frac{2}{5}R_3^1(x) + \frac{6}{35}R_5^1(x) + \frac{1}{35}R_7^1(x); \tag{24}$$

$m = 2$:

$$x^2 = R_2^2(x); \tag{25}$$

$$x^4 = \frac{3}{4}R_2^2(x) + \frac{1}{4}R_4^2(x); \tag{26}$$

$$x^6 = \frac{3}{5}R_2^2(x) + \frac{1}{3}R_4^2(x) + \frac{1}{15}R_6^2(x); \tag{27}$$

$$x^8 = \frac{1}{2}R_2^2(x) + \frac{5}{14}R_4^2(x) + \frac{1}{8}R_6^2(x) + \frac{1}{56}R_8^2(x). \tag{28}$$

Much of this work is based on the representation as a terminating hypergeometric function,

$$R_n^m(x) = \frac{n!x^n}{(\frac{n-m}{2})!(\frac{n+m}{2})!} F\left(-\frac{n-m}{2}, -\frac{n+m}{2}; -n; \frac{1}{x^2}\right), \tag{29}$$

with the three negative integer parameters

$$a \equiv -(n-m)/2; \quad b \equiv -(n+m)/2; \quad c \equiv -n, \tag{30}$$

and the argument

$$z \equiv \frac{1}{x^2}. \tag{31}$$

In these variables, the three-term recurrence reads [16, 9, 23]

$$\begin{aligned} &2n\,(a-1)\,(b-1)\,R_{n+2}^m(x) \\ &= (n+1)\left[2n(n+2)x^2 - m^2 - n(n+2)\right] R_n^m(x) - 2ab(n+2)R_{n-2}^m(x). \end{aligned} \tag{32}$$

$R_n^m(x)$ is also a product of x^m times a polynomial of degree $n - m$,

$$R_n^m(x) = (-1)^a \binom{(n+m)/2}{(n-m)/2} x^m \left[1 - \frac{a(b-1)}{m+1}x^2 + \frac{a(a+1)(b-1)(b-2)}{2(m+1)(m+2)}x^4 - \cdots\right], \tag{33}$$

which can be summarized in terms of Jacobi Polynomials

$$R_n^m(x) = (-1)^a x^m P_{-a}^{(m,0)}(1-2x^2) = \binom{n}{-a} x^m G_{-a}(m+1, m+1, x^2). \tag{34}$$

2.2. **Derivatives.** Derivatives of (29) are [1, 3.3.8]

$$\frac{d}{dx} R_n^m(x) = \binom{n}{(n-m)/2} \Big[\frac{d}{dx} x^n F(a,b;c;z) + x^n \frac{d}{dx} F(a,b;c;z) \Big]; \tag{35}$$

$$\frac{d^2}{dx^2} R_n^m(x) = \binom{n}{(n-m)/2} \Big[\frac{d^2}{dx^2} x^n F(a,b;c;z) + 2\frac{d}{dx} x^n \frac{d}{dx} F(a,b;c;z) + x^n \frac{d^2}{dx^2} F(a,b;c;z) \Big]; \tag{36}$$

$$\frac{d^3}{dx^3} R_n^m(x) = \binom{n}{(n-m)/2} \Big[\frac{d^3}{dx^3} x^n F(a,b;c;z) + 3\frac{d^2}{dx^2} x^n \frac{d}{dx} F(a,b;c;z) + 3\frac{d}{dx} x^n \frac{d^2}{dx^2} F(a,b;c;z) + x^n \frac{d^3}{dx^3} F(a,b;c;z) \Big]. \tag{37}$$

Faà di Bruno's Formula [10, 0.430.2][13] relegates the derivatives w.r.t. x to derivatives w.r.t. z,

$$\frac{d}{dx} F(a,b;c;z) = -2x^{-3} F'(a,b;c;z);$$

$$\frac{d^2}{dx^2} F(a,b;c;z) = 6x^{-4} F'(a,b;c;z) + 4x^{-6} F''(a,b;c;z);$$

$$\frac{d^3}{dx^3} F(a,b;c;z) = -24x^{-5} F'(a,b;c;z) - 36x^{-7} F''(a,b;c;z) - 8x^{-9} F'''(a,b;c;z).$$

After insertion of these three formulas into (35)–(37), the derivatives of $R_n^m \cong x^n F$ are

$$R_n^{m\prime}(x) \cong nx^{n-1}F - 2x^{n-3}F'; \tag{38}$$

$$R_n^{m\prime\prime}(x) \cong n(n-1)x^{n-2}F - 2(2n-3)x^{n-4}F' + 4x^{n-6}F''; \tag{39}$$

$$R_n^{m\prime\prime\prime}(x) \cong n(n-1)(n-2)x^{n-3}F - 6(n-2)^2 x^{n-5} F' + 12(n-3)x^{n-7}F'' - 8x^{n-9}F''', \tag{40}$$

where $\cong$ means the binomial factor and the argument list $(a,b;c;z)$ of the hypergeometric function have not been written down explicitly. Since $R_n^m(x)$ is a polynomial of order n, the $(n+1)$st derivatives equal zero. Backward elimination of F and its derivatives with the aid of [1, 15.5.1]

$$z(1-z)F''(a,b;c;z) + [c-(a+b+1)z]F'(a,b;c;z) = abF(a,b;c;z) \tag{41}$$

leads to the analog of (2),

$$x^2(x^2-1)\frac{d^2}{dx^2} R_n^m(x) = \left[nx^2(n+2) - m^2\right] R_n^m(x) + x(1-3x^2)\frac{d}{dx} R_n^m(x), \tag{42}$$

which is one special case of differential equations that generate orthogonal functions [19], and could also be obtained by applying the derivatives of [1, 22.6.1] [5, 3] to (34). The derivative of this reaches out to the third derivatives, in which R'' is

reduced to R and R' with the help of the previous equation,

$$x^3(x^2-1)^2\frac{d^3}{dx^3}R_n^m(x) = \left[x^2(n^2+2n+7m^2) - 5x^4n(2+n) - 3m^2\right]R_n^m(x) \tag{43}$$
$$+x\left[6x^2(2x^2-1) - m^2(x^2-1) + 2 + x^2(x^2-1)n(n+2)\right]\frac{d}{dx}R_n^m(x).$$

2.3. **Zeros.**

2.3.1. *Ratios of Derivatives.* Installation of f/f' in (1) progresses by dividing $R_n^m \cong x^n F$ through (38),

$$\frac{R_n^m(x)}{R_n^{m\prime}(x)} = \frac{x}{n - 2z\frac{F'(a,b;c;z)}{F(a,b;c;z)}}. \tag{44}$$

The analog of (5) is implemented by substituting [1, 15.2.1]

$$F'(a,b;c;z) = \frac{ab}{c}F(a+1,b+1;c+1;z) \tag{45}$$

in the denominator. In lieu of (7) we find the continued fractions [7]

$$\frac{F(a,b;c;z)}{F(a+1,b+1;c+1;z)} \equiv \frac{-bz}{c} + 1 - \frac{\frac{(a+1)(c-b)z}{c(c+1)}}{\frac{(a+1-b)z}{c+1} + 1 - \cdots} \frac{\frac{(a+2)(c+1-b)z}{(c+1)(c+2)}}{\frac{(a+2-b)z}{c+2} + 1 - \cdots} \tag{46}$$

which terminate in our cases since a is a negative integer and $c = a + b$. This already suffices to implement the standard Newton iteration, ie, to approximate (1) by $\Delta x = -f(x)/f'(x)$. Division of (42) through $R_n^{m\prime}(x)$ yields

$$\frac{R_n^{m\prime\prime}(x)}{R_n^{m\prime}(x)} = \frac{1}{x^2-1}\left[\left(n(n+2) - \frac{m^2}{x^2}\right)\frac{R_n^m(x)}{R_n^{m\prime}(x)} + \frac{1-3x^2}{x}\right]. \tag{47}$$

This is $f''(x)/f'(x)$ of the generic formula, and can be quickly computed from $R_n^m(x)/R_n^{m\prime}(x) = f(x)/f'(x)$ of the lower order.

2.3.2. *Initial Guesses.* For n and m fixed, the strategy adopted here is to compute the $(n-m)/2$ distinct roots in $(0,1)$ starting with the smallest, then bootstrapping the others in naturally increasing order. An approximation to the smallest root is found by equating the first three terms in the square bracket of (33) with zero—hoping that higher powers of x become insignificant for small x—and solving the bi-quadratic equation for x. This guess may become unstable for n approximately larger than 11 in the sense that the Newton iterations converge to another than this smallest root. Instead, the simple, heuristic initial guess

$$x \approx \frac{1.46m + 2.41}{n + 0.46m + 1.06} \tag{48}$$

is used in general, but keeping the solution to the bi-quadratic equation when this is exact, ie, in the cases $n - m = 2$ or 4.

A shooting method is useful to produce an initial estimate for one root supposed an adjacent one has already been found. The third order Taylor extrapolation from one root x to the next one at $x + \Delta x$ is

$$f(x+\Delta x) \approx f(x) + \Delta x f'(x) + \frac{(\Delta x)^2}{2!}f''(x) + \frac{(\Delta x)^3}{3!}f'''(x) \approx 0. \tag{49}$$

Division through $\Delta x f'$ and exploiting $f(x) = 0$ yields a quadratic equation for the approximate distance Δx to the next one,

$$1 + \frac{\Delta x}{2}\frac{f''(x)}{f'(x)} + \frac{(\Delta x)^2}{6}\frac{f'''(x)}{f'(x)} \approx 0, \tag{50}$$

from which the branch $\Delta x > 0$ is systematically selected to start computation of the root adjacent to the previous one. The two ratios of derivatives are obtained by setting $R_n^m(x) = 0$ in (42) and (43), then dividing both equations through $R_n^{m\prime}(x)$. This aim to locate the next root with sufficient accuracy—and to prevent the Newton's Method to be be drawn into the second next root which would call for more administrative care [4]—is the rationale to look into third derivatives; it might also guide the way to even higher order Newton's methods employing f'''.

3. Summary

The Newton's Method of third order convergence is implemented for Zernike Polynomials R_n^m by computation of the ratios $R_n^{m\prime\prime}/R_n^{m\prime}$ and $R_n^m/R_n^{m\prime}$ with relay to the generic formulas of associated, terminating hypergeometric functions. Adding knowledge on the derivative $R_n^{m\prime\prime\prime}$, a shooting method is proposed which generates an initial guess for the adjacent root from each root found.

Appendix A. Table of Roots of Low-order Polynomials

The roots $x_{i,n,m}$ of $R_n^m(x)$ are tabulated below for $2 \le n \le 20$ in two major columns. Each column contains n, then m, then $(n-m)/2 = |a|$ values of $x_{i,n,m}$. Only the roots $x > 0$ are included, and only the standard parameter range for even, positive values of $n - m$ is considered.

Because the $G_{-a}(m+1, m+1, y)$ mentioned in (34) build a system of orthogonal polynomials with weight y^m over the unit interval $0 \le y \le 1$, the squares $x_{i,n,m}^2 = y_{i,n,m}$ are also the abscissae for Gaussian integration of moment m [1, Tab. 25.8][6].

2	0	0.7071067811865475727	14	4	0.8246570394661102421
4	0	0.4597008433809830485	14	4	0.9278396109654096779
4	0	0.8880738339771152567	14	4	0.9862121569592748882
6	0	0.3357106870197287818	16	4	0.4352810401596804435
6	0	0.7071067811865475727	16	4	0.6105335011400015999
6	0	0.9419651451198933767	16	4	0.7533739926853023627
8	0	0.2634992299855423159	16	4	0.8653533945767831748
8	0	0.5744645143153508382	16	4	0.9446613988337823065
8	0	0.8185294874300058643	16	4	0.9894368252983534173
8	0	0.9646596061808674349	18	4	0.3914606479189805532
10	0	0.2165873427295972042	18	4	0.5534677249323020076
10	0	0.4803804169063914387	18	4	0.6903188343084598610
10	0	0.7071067811865475727	18	4	0.8039833969179427386
10	0	0.8770602345636481223	18	4	0.8931498697389127495
10	0	0.9762632447087885579	18	4	0.9561484720295934103
12	0	0.1837532119404283737	18	4	0.9916369448881094950
12	0	0.4115766110542091183	20	4	0.3554897669109032265
12	0	0.6170011401597257361	20	4	0.5054988151106132310
12	0	0.7869622564275865484	20	4	0.6353277098425955671
12	0	0.9113751660173390334	20	4	0.7472059125768437671
12	0	0.9829724091252897145	20	4	0.8402437401628405356
14	0	0.1595181614381909196	20	4	0.9130561549207910632

14	0	0.3594918736220650279
14	0	0.5450480935764305812
14	0	0.7071067811865475727
14	0	0.8384047803350709316
14	0	0.9331482158798232174
14	0	0.9871949939963123866
16	0	0.1409080258581174028
16	0	0.3188522562146716699
16	0	0.4870665201405610101
16	0	0.6389700139694938219
16	0	0.7692316434259740543
16	0	0.8733648750425931917
16	0	0.9478044306220632098
16	0	0.9900226907746954019
18	0	0.1261740078074202742
18	0	0.2863292621034079777
18	0	0.4396752024502914580
18	0	0.5812686885581361818
18	0	0.7071067811865475727
18	0	0.8137116883158951319
18	0	0.8981568439589463493
18	0	0.9581312820607194025
18	0	0.9920081248426411147
20	0	0.1142223084227163565
20	0	0.2597466393536357887
20	0	0.4003688497504367394
20	0	0.5322614986408245041
20	0	0.6523517689936806363
20	0	0.7579163340968551044
20	0	0.8465800003925344486
20	0	0.9163540713880810040
20	0	0.9656768006659848247
20	0	0.9934552150241026114
3	1	0.8164965809277260345
5	1	0.5958615826865180098
5	1	0.9192110607898046348
7	1	0.4608042298407784010
7	1	0.7684615381131740808
7	1	0.9546790248493448594
9	1	0.3738447061866471688
9	1	0.6452980455813291938
9	1	0.8503863747508400017
9	1	0.9710282199223060351
11	1	0.3139029878781443572
11	1	0.5518475574344458012
11	1	0.7496833930084177977
11	1	0.8955370355972955831
11	1	0.9798929242261785744
13	1	0.2702856427564344077
13	1	0.4803812423169180335
13	1	0.6643255837527634045
13	1	0.8142575205172167818
13	1	0.9229958831606540626
20	4	0.9643628549735204780
20	4	0.9932085369576263423
7	5	0.9258200997725514192
9	5	0.7942238940183964369
9	5	0.9616464847987593600
11	5	0.6840930020506119646
11	5	0.8727107021799724862
11	5	0.9758129824224018867
13	5	0.5974058327888663866
13	5	0.7840085394020771536
13	5	0.9116353458812981314
13	5	0.9831524024611432155
15	5	0.5288602129223232140
15	5	0.7061939018640647214
15	5	0.8404366834692439392
15	5	0.9345601788393610443
15	5	0.9875190300435001678
17	5	0.4737815904024170188
17	5	0.6399479279255915198
17	5	0.7731681889583568168
17	5	0.8765920400095927878
17	5	0.9493949303895553582
17	5	0.9903520628752460198
19	5	0.4287525723551651180
19	5	0.5837872858381362162
19	5	0.7126994626270685140
19	5	0.8187572351505322255
19	5	0.9014368896323405878
19	5	0.9596126391703136971
19	5	0.9923041120472186893
8	6	0.9354143466934853324
10	6	0.8164965809277260345
10	6	0.9660917830792958849
12	6	0.7132667548676191460
12	6	0.8854995128634446377
12	6	0.9783359081211411290
14	6	0.6297279581530694781
14	6	0.8030111569365681046
14	6	0.9197999176484970008
14	6	0.9847470483288770504
16	6	0.5622633194744699470
16	6	0.7291002958196058925
16	6	0.8535798917458220503
16	6	0.9401303969875089983
16	6	0.9885994756607009437
18	6	0.5071545248007354179
18	6	0.6650990116151613840
18	6	0.7899637876841181239
18	6	0.8860816722252334854
18	6	0.9533796195297132847
18	6	0.9911212731459003722
20	6	0.4615059331053839586
20	6	0.6100885865453433698

13	1	0.9852327505925770890	20	6	0.7319787736696559133
15	1	0.2371973029714337655	20	6	0.8314254884667372503
15	1	0.4245476318823276363	20	6	0.9085207085711664021
15	1	0.5938221258198196351	20	6	0.9625678871955013483
15	1	0.7396983346814803850	20	6	0.9928726539379972849
15	1	0.8568606521572865731	9	7	0.9428090415820633563
15	1	0.9409149519691435426	11	7	0.8343946751715023424
15	1	0.9886964213353295339	11	7	0.9696128930707756943
17	1	0.2112674970031354627	13	7	0.7374505105030352281
17	1	0.3799555987772065824	13	7	0.8959410734924830866
17	1	0.5355602273735704522	13	7	0.9803818873685179458
17	1	0.6743984839815206911	15	7	0.6571568797605316092
17	1	0.7925073093774768207	15	7	0.8189060737060670503
17	1	0.8863924190810575920	15	7	0.9265772077286310582
17	1	0.9532451899171618948	15	7	0.9860657631628138020
17	1	0.9910701715078688023	17	7	0.5911284719636833440
19	1	0.1904148441776781775	17	7	0.7486326613042562084
19	1	0.3436262195904129513	17	7	0.8647030262108379439
19	1	0.4870081978929722277	17	7	0.9448234063345077871
19	1	0.6179666376955199603	17	7	0.9895076756006223606
19	1	0.7335050094709572033	19	7	0.5364117760083304542
19	1	0.8308624555679250401	19	7	0.6868800285074768697
19	1	0.9076801990993329516	19	7	0.8044060078556206639
19	1	0.9620876312949896425	19	7	0.8942045500841436789
19	1	0.9927676416720699892	19	7	0.9567806932331796022
4	2	0.8660254037844385966	19	7	0.9917768363310000002
6	2	0.6751652804971347566	10	8	0.9486832980505137680
6	2	0.9367417879781805290	12	8	0.8490975736565613552
8	2	0.5431369901889407936	12	8	0.9724710674756380513
8	2	0.8080818238035354373	14	8	0.7578370705278929531
8	2	0.9628114955311087853	14	8	0.9046299747290316162
10	2	0.4518280448392144044	14	8	0.9820745595413366003
10	2	0.6949479871872660253	16	8	0.6807458887789931135
10	2	0.8725819517089140609	16	8	0.8324053937687126981
10	2	0.9754483331027854476	16	8	0.9322943792644380334
12	2	0.3859349518416070879	16	8	0.9871744960883001019
12	2	0.6047036690554417060	18	8	0.6163473579853351314
12	2	0.7810976974325815059	18	8	0.7654979183523115127
12	2	0.9091312772247495122	18	8	0.8742424706423237435
12	2	0.9825584257853499093	18	8	0.9488319407257733706
14	2	0.3364437305441106418	18	8	0.9902817948430912010
14	2	0.5332155968164153936	20	8	0.5623002252838306125
14	2	0.7006879382687045688	20	8	0.7059431502741818631
14	2	0.8352024197624841051	20	8	0.8169643175952393532
14	2	0.9318991674757601817	20	8	0.9012382495289457118
14	2	0.9869627373443907725	20	8	0.9597179665033356288
16	2	0.2980215318345257325	20	8	0.9923422148430279810
16	2	0.4759020418201234115	11	9	0.9534625892455923513
16	2	0.6324385240645979955	13	9	0.8613939328976448762
16	2	0.7655049675886762550	13	9	0.9748375170636489617
16	2	0.8714619176716458249	15	9	0.7752632451107936973
16	2	0.9470538373566472767	15	9	0.9119748762120825081
16	2	0.9898824305070776930	15	9	0.9834982084062168228

18	2	0.2673780750923551164
18	2	0.4292182386813452322
18	2	0.5748458299193258680
18	2	0.7031379109502706726
18	2	0.8113864729725273062
18	2	0.8969530821125818454
18	2	0.9576530595755170516
18	2	0.9919184646169658670
20	2	0.2423925241972734457
20	2	0.3905933409401172729
20	2	0.5260526858211680423
20	2	0.6483100675208097741
20	2	0.7553451890441794658
20	2	0.8450496504116072893
20	2	0.9155547407096559231
20	2	0.9653576903957581390
20	2	0.9933952458125041574
5	3	0.8944271909999158554
7	3	0.7279134123608967943
7	3	0.9480050066727199187
9	3	0.6027143852742457009
9	3	0.8359493221264154839
9	3	0.9684648164078416555
11	3	0.5113489892733628084
11	3	0.7320153318669290199
11	3	0.8889787567592866147
11	3	0.9786966233548161087
13	3	0.4429582456583350258
13	3	0.6458329596901977165
13	3	0.8053384408042754128
13	3	0.9195679148240427647
13	3	0.9845992603558400003
15	3	0.3902219391376385849
15	3	0.5755870542650849409
15	3	0.7296653799715601130
15	3	0.8518232083769392560
15	3	0.9389444522639546209
15	3	0.9883295764481713208
17	3	0.3484639562887706932
17	3	0.5180380683998889735
17	3	0.6639777561205169043
17	3	0.7865138450349050681
17	3	0.8833196205032936010
17	3	0.9520307422269438380
17	3	0.9908430322275690871
19	3	0.3146454476233096487
19	3	0.4703652913957364068
19	3	0.6075556764710252633
19	3	0.7270113006012387524
19	3	0.8270388433632478442
19	3	0.9056952852248679742
19	3	0.9612979313921893310
19	3	0.9926194840347464243
17	9	0.7012604730894272942
17	9	0.8440171326522077910
17	9	0.9371828916889872740
17	9	0.9881197223055366852
19	9	0.6385852750050469151
19	9	0.7802151464898949840
19	9	0.8825163921882515083
19	9	0.9522959699168392911
19	9	0.9909494948025596717
12	10	0.9574271077563381027
14	10	0.8718317153731350855
14	10	0.9768291428674981125
16	10	0.7903347449834110527
16	10	0.9182661541320223941
16	10	0.9847122732288815516
18	10	0.7192722131418530784
18	10	0.8541140545067785750
18	10	0.9414112100240797920
18	10	0.9889351394546163077
20	10	0.6583509748272358131
20	10	0.7931750687612736384
20	10	0.8897625152667285597
20	10	0.9553196436506564693
20	10	0.9915313152497930993
13	11	0.9607689228305228424
15	11	0.8808037886787085657
15	11	0.9785284968679074380
17	11	0.8035014494300847243
17	11	0.9237159298241287564
17	11	0.9857598779860213822
19	11	0.7352174806998263978
19	11	0.8629761939750234534
19	11	0.9451049233245847336
19	11	0.9896457777687525104
14	12	0.9636241116594315148
16	12	0.8885993155618584494
16	12	0.9799955389607631906
18	12	0.8151048281585070443
18	12	0.9284828318578448592
18	12	0.9866730715175761057
20	12	0.7494357022184537920
20	12	0.8708181789128337513
20	12	0.9483595986475444883
20	12	0.9902706191583880990
15	13	0.9660917830792958849
17	13	0.8954359308455520639
17	13	0.9812748728586200286
19	13	0.8254089266585322715
19	13	0.9326878942873719769
19	13	0.9874761693913842731
16	14	0.9682458365518542553
18	14	0.9014806129319800077
18	14	0.9824003676126956686

6	4	0.9128709291752769017	20	14	0.8346212400166618250
8	4	0.7657261797294159233	20	14	0.9364250918976404492
8	4	0.9558574253919850383	20	14	0.9881879542766879299
10	4	0.6481612911435379321	17	15	0.9701425001453318764
10	4	0.8566734238949372804	19	15	0.9068636367673578169
10	4	0.9726240720110040927	19	15	0.9833982044520968024
12	4	0.5586908741397313971	18	16	0.9718253158075500497
12	4	0.7608650731649617693	20	16	0.9116881483483436632
12	4	0.9016049670538779370	20	16	0.9842889413994087011
12	4	0.9811849966449255334	19	17	0.9733285267845752653
14	4	0.4896856591758441124	20	18	0.9746794344808964450
14	4	0.6789209633908173114			

Appendix B. PARI implementation

The full source code of the PARI interpreter program which computed the values shown in Appendix A is listed below. The language is similar to C/C++ and has inherent support for arbitrary precision computation [22].

`HypergAugmRatio` implements (46). `HypergRatio` implements (45). `ZernikePrratio` implements (44). `Zernike2Prratio` implements (47). `Zernike3Prratio` implements (43). `ZernikeRoot` implements (1). `ZernikeRootEst` implements (48), but (33) if $n = m + 2$. `ZernikeAllRoot` implements a loop with guesses as in (50). `main` loops over n and m to tabulate the zeros up to a maximum n.

```
/** Compute the quotient of a Gauss Hypergeometric Function over
* the Function with the same argument but all the three parameters
* increased by 1.
* @param[in] a first parameter of the Gauss Hypergeometric Function F(a,b;c;z).
*     This must be a negative integer.
* @param[in] b second parameter of F
* @param[in] c third parameter of F
* @param[in] z argument of F
* @return the ratio F(a,b;c;z)/F(a+1,b+1;c+1;z)
* @warning the function assumes that the parameter a is a negative integer
*/
HypergAugmRatio(a,b,c,z)={
    local(ff=0.0) ;
    forstep(ap2= -1,a+1,-1,
        ff = z*(ap2-b)+(ap2+c-a)*(1.0-ff) ;
        ff = ap2*(ap2+c-a-b-1)*z/((ap2+c-a-1)*ff) ;
    ) ;
    return(-b*z/c+1-ff) ;
}

/** Compute the quotient of a Gauss Hypergeometric Function over
* its first derivative.
* @param[in] a first parameter of the Gauss Hypergeometric Function F(a,b;c;z).
*     This must be a negative integer.
* @param[in] b second parameter of F
* @param[in] c third parameter of F
* @param[in] z argument of F
* @return ratio of the value divided by the first derivative, F/F'.
* @warning Checking a against being a negative integer is not done.
```

```
*/
HypergRatio(a,b,c,z)={
    c*HypergAugmRatio(a,b,c,z)/(a*b) ;
}

/** Compute the ratio of the Zernike polynomial over its first derivative.
* @param[in] n first parameter of the Zernike polynomial, a positive integer
* @param[in] m second parameter of the polynomial, a positive integer
*      less than or equal to n, with n-m even.
* @param[in] x the argument of the polynomial in the interval [0,1]
* @param[in] ffprime the ratio F'(a,b;c;z)/F(a,b;c;z) of the Gauss
*     hypergeometric function, where a=-(n-m)/2, b=-(n+m)/2, c=a+b,
*     and z=1/x^2.
* @return the ratio R/R'
* @warning no check is done that n-m is a positive even integer
*  or that n and m are individually positive or that x is between 0
*  and one.
*/
ZernikePrratio(n,m,x,ffprime) = {
    local(z=1.0/x^2) ;
    /** To enhance stability in the case of ffprime close to
    * zero, we do not use the equivalent x/(n-2*z/ffprime) ;
    */
    x*ffprime/(n*ffprime-2*z) ;
}

/** Compute the ratio of the second derivative of the Zernike polynomial
*  over the first derivative, with respect to the argument x.
* @param[in] n first parameter of the Zernike polynomial, a positive integer
* @param[in] m second parameter of the polynomial, a positive integer
*      less than or equal to n, with n-m even.
* @param[in] x the argument of the polynomial in the interval [0,1]
* @param[in] rrprime the ratio R(x)/R'(x)
* @return the ratio of derivatives R''/R'
* @warning no checking is done that n-m is a positive even integer
*  or that n and m are individually positive or that x is between 0
*  and one.
*/
Zernike2Prratio(n,m,x,rrprime) = {
    local(xsq=x^2) ;
    ((n*(n+2)*xsq-m^2)*rrprime+x*(1-3*xsq))/(xsq*(xsq-1))  ;
}

/** Compute the ratio of the third derivative of the Zernike polynomial
*  over the first derivative. Derivatives are with respect to the argument x.
*  The ratio R'/R of the first derivative is to be provided.
* @param[in] n first parameter of the Zernike polynomial, a positive integer
* @param[in] m second parameter of the polynomial, a positive integer
*      less than or equal to n, with n-m even.
* @param[in] x the argument of the polynomial in the interval [0,1]
* @param[in] rrprime the ratio R'/R
* @return the ratio R'''/R'
* @warning no test is performed that n-m is a positive even integer
```

```
*  or that n and m are individually positive or that x is between 0
*  and one.
*/
Zernike3Prratio(n,m,x,rrprime) = {
    local(x2=x^2,x2m=x^2-1, n2n=n*(n+2), m2=m^2) ;
    ( (x2*(n2n+7*m2)-5*x2^2*n2n-3*m2) *rrprime
            +x*(6*x2*(2*x2-1)-m2*x2m+2+x2*x2m*n2n) )
        /(x*x2*x2m^2) ;
}

/** Compute a root of the Zernike polynomial within a specified error bar.
* @param[in] n first parameter of the Zernike polynomial, a positive integer
* @param[in] m second parameter of the polynomial, a positive integer
*      less than or equal to n, with n-m even.
* @param[in] x an initial guess of the root
* @param[in] eps the absolute accuracy of the result. The Newton iteration
*      will be terminated if two subsequent estimates agree within this limit.
* @return a root x such that R(x)=0
* @warning no test is performed that n-m is a positive even integer
*  or that n and m are individually positive or that x is between 0
*  and one.
*/
ZernikeRoot(n,m,x,eps) = {
    /** The variable root is used to keep a history of the most recent
    * approximation to the root. The variables a, b, and c are the
    * corresponding arguments to the associated Gauss Hypergeometric Function.
    */
    local(root=x,z, a= -(n-m)/2, b= -(n+m)/2, c= -n, fprimef,ffprime,rrp,r2prp) ;
    if(n==m,
        0.0,

        /** For an absolute accuracy of 1.e-30, up to 10 iterations
        * will be needed for n<=10. So we update the root with up to a
        * maximum of 20 Newton iterations.
        */
        for(i=1,20,
            z=1/root^2 ;

            /* ffprime contains F'(a,b;c;z)/F(a,b;c;z) of the
            * associated Hypergeometric function.
            */
            ffprime= HypergRatio( a, b, c, 1/root^2) ;

            /* rrp and r2prp are the ratios R/R' and R''/R' of
            * the Zernike polynomial relative to its 1st and 2nd derivative.
            * This completes the relay F'/F -> R/R' -> R''/R'.
            */
            rrp= ZernikePrratio(n,m,x,ffprime) ;
            r2prp= Zernike2Prratio(n,m,x,rrp) ;

            x=root ;
            /** Now perform the third order Newton update. The formula
            * is Delta(x) = -(f/f')/[1-(f/f')*(f''/f')/2]. Reduction to the
```

```
            * 2nd order would be implemented as the simpler
            * root = x-rrp ;
            */
            root = x-rrp/(1.0-0.5*rrp*r2prp) ;

            /* terminate if the old and the new guess agree within eps.
            */
            if( abs(x-root) < eps,
                break ;
            ) ;
        ) ;
        return(root) ;
    ) ;
} /* ZernikeRoot */

/** Provide a guess of the smallest nonzero root of the Zernike Polynomial.
* @param[in] n first parameter of the Zernike polynomial, a positive integer
* @param[in] m second parameter of the polynomial, a positive integer
*      less than or equal to n, with n-m even.
* @return an estimate of a root which is the smallest nontrivial (nonzero)
*   positive between 0 and 1 out of the total of (n-m)/2 .
*/
ZernikeRootEst(n,m) = {
    local(a= -(n-m)/2, b= -(n+m)/2, x2) ;
    if(n==m,
        0,
        if(n == m+2,
            /** if n equals m+2, the resulting equation is
            * a linear equation in x^2.
            */
            x2 = (m+1)/(a*(b-1)) ;
            return( sqrt(x2)) ,

            /* if n does not equal m+2, a simple heuristic estimate
            * with a rational function of n and m is used.
            */
            return((1.46*m+2.41)/(n+0.46*m+1.06)) ;
        ) ;
    ) ;
} /* ZernikeRootEst */

/**
* @param[in] n first parameter of the Zernike polynomial, a positive integer
* @param[in] m second parameter of the polynomial, a positive integer
*      less than or equal to n, with n-m even.
* @param[in] eps the desired absolute accuracy of each root.
* @return the (n-m)/2 positive roots in the open interval from 0 to 1
* @note the m-fold degenerate root at zero is not returned, nor the
*   symmetric values on the negative real axis.
* @warning no test is performed that n-m is a positive even integer
*  or that n and m are individually positive or that x is between 0
*  and one.
*/
```

```
ZernikeAllRoot(n,m,eps) = {
    local(s=(n-m)/2,rs,x,r2pr,r3pr,disc) ;

    /* The result contains s=(n-m)/2 individual values, which are
    * collected in the vector rs.
    */
    rs=vector(s) ;
    for(i=1,s,
        if(i==1,
            /** If this is the first root for a pair (n,m),
            * we attach to the leftmost (positive, but smallest in value)
            * of them, and call ZernikeRootEst() for an initial value.
            */
            x=ZernikeRootEst(n,m),

            /** Otherwise, for the 2nd and higher roots, we take the
            * previous root, and solve the quadratic equation for the step
            * D to the next root which follows from a 3rd order Taylor
            * approximation at the known root. This estimate is then
            * used to call the Newton routine for the next root.
            */
            x=rs[i-1] ;

            /** Compute R''/2R' at the old root. The general formula
            * x^2(x^2-1) [R''/R'] = {n(n+2)x^2-m^2}*[R/R']+x(1-3x^2)
            * is simplified because R and therefore R/R' are known
            * to be zero for the recent root.
            */
            r2pr= Zernike2Prratio(n,m,x,0)/2 ;

            /** Compute R'''/6R' at the old root. Application of the formula
            * x^3(x^2-1)^2 [R'''/R']
            * = {x^2(n^2+2n+7m^2...}*[R/R']+x*[6x^2(2x^2-1)-m^2(x^2-1)+...]
            * uses that R and therefore R/R' are known
            * to be zero for the recent root.
            */
            r3pr= Zernike3Prratio(n,m,x,0)/6 ;

            /* The quadratic estimate from 1+(r2pr/2)*D+(r3pr/6)*D^2=0
            * is in these variables 1+r2pr*D+r3pr*D^2=0, after division
            * 1/r3pr + r2pr/r3pr*D +D^2 = 0. disc is the discriminant
            * of the quadratic.
            */
            disc = 1-4*r3pr/r2pr^2 ;

            /* This third order local approximation to the full polynomial
            * usually has a root to the left and another one to the right of
            * the current root. We select the sign of the square root of
            * the discriminant to lock into the larger of these two values.
            */
            if(disc>0,
                if( r2pr/r3pr>0,
                    x += r2pr/(2*r3pr)*(-1+sqrt(disc)) ,
```

```
                    x += r2pr/(2*r3pr)*(-1-sqrt(disc)) ;
                ) ,
                /** If the discriminant was negative, just ignoring the
                * square root works also and keeps arithmetics real-valued.
                */
                x -= r2pr/(2*r3pr) ;
            ) ;
        ) ;
        /** Given the starting value x of the next root, call the Newton
        * routine to converge to the root within eps, and store the result
        * as another value in the vector.
        */
        rs[i] = ZernikeRoot(n,m,x,eps) ;
    ) ;
    return(rs) ;
} /* ZernikeAllRoot */

/** The main routine tabulates the zeros of the Zernike Polynomials
* of lowest order.
* @param[in] nmax the maximum value of the parameter n to be used
* @return 0
*/
main(nmax)={
    /** we loop with m and n over all cases that have non-zero roots.
    */
    for(m=0,nmax,
        /* The parameter n runs from m up in steps of 2.
        */
        forstep(n=m,nmax,2,
            /* Collect the roots of this (n,m) with an accuracy of 1.e-30
            * in the vector rts.
            */
            rts=ZernikeAllRoot(n,m,1.e-30) ;

            /* Print the (n-m)/2 results. */
            for(i=1,(n-m)/2,
                print(n" "m" "rts[i]) ;
            ) ;
        ) ;
    ) ;
    return(0) ;
}  /* main */

{
    /** Collect all non-trivial roots up to some maximum explicit n.
    */
    main(20) ;
}
```

References

1. Milton Abramowitz and Irene A. Stegun (eds.), *Handbook of mathematical functions*, 9th ed., Dover Publications, New York, 1972. MR 0167642 (29 #4914)

2. G. Conforti, *Zernike aberration coefficients from Seidel and higher-order power-series coefficients*, Opt. Lett. **8** (1983), no. 7, 407–408.
3. E. H. Doha, *On the coefficients of differentiated expansions and derivatives of Jacobi polynomials*, J. Phys. A: Math. Gen. **35** (2002), 3467–3478. MR 1907373 (2003e:33020)
4. L. W. Ehrlich, *A modified Newton method for polynomials*, Commun. ACM **10** (1967), no. 2, 107–108.
5. David Elliott, *Uniform asymptotic expansions of the Jacobi polynomials and an associated function*, Math. Comp. **25** (1971), no. 114, 309–315.
6. Herbert Fishman, *Numerical integration constants*, Math. Tables Aids Comp. **11** (1957), no. 57, 1–9. MR 0086391 (19,177g)
7. Evelyn Frank, *A new class of continued fraction expansions for the ratios of hypergeometric functions*, Trans. Am. Math. Soc. **81** (1956), no. 2, 453–476.
8. Jürgen Gerlach, *Accelerated convergence in Newton's method*, SIAM Review **36** (1994), no. 2, 272–276. MR 1278637 (95e:65053)
9. Amparo Gil, Javier Segura, and Nico M. Temme, *Numerically satisfactory solutions of hypergeometric recursions*, Math. Comp. **76** (2007), no. 259, 1449–1468.
10. I. Gradstein and I. Ryshik, *Summen-, Produkt- und Integraltafeln*, 1st ed., Harri Deutsch, Thun, 1981. MR 0671418 (83i:00012)
11. Eldon Hansen and Merrell Patrick, *A family of root finding methods*, Numer. Math. **27** (1977), no. 3, 257–269. MR 0433858 (55 #6829)
12. D. J. Hofsommer, *Note on the computation of the zeros of functions satisfying a second order differential equation*, Math. Tabl. Aids Comput. **12** (1958), no. 61, 58–60, E: [20, (C2)]. MR 0099752 (20 #6190)
13. Warren P. Johnson, *The curious history of Faà di Bruno's formula*, Am. Math. Monthly **109** (2002), no. 3, 217–234. MR 1903577 (2003d:01019)
14. Bahman Kalantari, Iraj Kalantari, and Rahim Zaare-Nahandi, *A basic family of iteration functions for polynomial root finding and its characterizations*, J. Comp. Appl. Math. **80** (1997), no. 2, 209–226. MR 1455244 (98d:65066)
15. Eric C. Kintner, *On the mathematical properties of the Zernike polynomials*, Optica Acta **23** (1976), no. 8, 679–680.
16. Eric C. Kintner and Richard M. Sillitto, *A new analytic method for computing the optical transfer functions*, Optica Acta **23** (1976), no. 8, 607–619.
17. Stanisław Lewanowicz, *Recurrences for the coefficients of series expansions with respect to classical orthogonal polynomials*, Applic. Mathemat. **29** (2002), no. 1, 97–116. MR 1907630 (2003d:33017)
18. Francisco Marcellán and Ridha Sfaxi, *Second structure relation for semiclassical orthogonal polynomials*, J. Comput. Appl. Math. **200** (2007), no. 2, 537–554. MR 2289233
19. Mohammad Masjed-Jamei, *A basic class of symmetric orthogonal polynomials using the extended Sturm-Liouville theorem for symmetric functions*, J. Math. Anal. Appl. **325** (2007), no. 2, 753–775. MR 2270049
20. Richard J. Mathar, *Numerical representation of the incomplete gamma function of complex argument*, arXiv:math.NA/0306184 (2003).
21. Robert J. Noll, *Zernike polynomials and atmospheric turbulence*, J. Opt. Soc. Am. **66** (1976), no. 3, 207–211.
22. The PARI-Group, Bordeaux, *PARI/GP, version 2.3.2*, 2007, available from `http://pari.math.u-bordeaux.fr/`.
23. Aluizio Prata, Jr. and W. V. T. Rusch, *Algorithm for computation of Zernike polynomials expansion coefficients*, Appl. Opt. **28** (1989), no. 4, 749–754.
24. T. S. Shao, T. C. Chen, and R. M. Frank, *Tables of zeros and Gaussian weights of certain associated Laguerre polynomials and the related generalized Hermite polynomials*, Math. Comp. **18** (1964), no. 88, 598–616. MR 0166397 (29 #3674)
25. William J. Tango, *The circle polynomials of Zernike and their application in optics*, Appl. Phys. A **13** (1977), no. 4, 327–332.
26. Robert K. Tyson, *Conversion of Zernike aberration coefficients to Seidel and higher-order power series aberration coefficients*, Opt. Lett. **7** (1972), no. 6, 262–264.

URL: `http://www.strw.leidenuniv.nl/~mathar`
E-mail address: `mathar@strw.leidenuniv.nl`

Leiden Observatory, Leiden University, P.O. Box 9513, 2300 RA Leiden, The Netherlands